Advances in Science and Technology
Research Journal
Volume 8, No. 22, June 2014, pp. 42–50
DOI: 10.12913/22998624.1105164

Research Article

SELF-LEARNING SCORING MODELS – INTRODUCTION OF AN ON-LINE APPROACH TO RISK ASSESMENT

Ryszard Kozera[1], Przemyslaw Koziol[2]

[1] Faculty of Applied Informatics and Mathematics, Warsaw University of Life Sciences (SGGW), Nowoursynowska 159, 02-776 Warsaw, Poland, e-mail: ryszard_kozera@sggw.edu.pi; ryszard.kozera@gmail.com

[2] Faculty of Mathematics and Information, Warsaw University of Technology, PL Politechniki 1, 00-661 Warsaw, Poland, e-mail: pr.koziol@gmail.com

Received: 2014.04.30
Accepted: 2014.05.20
Published: 2014.06.05

ABSTRACT
The problem considered in this article involves the construction of evaluation model, which could subsequently be used in the field of modeling and risk management. The research work is finalized by a construction of a new model on the basis of observations of the models used for risk management and knowledge of information theory, machine learning and artificial neural networks. The developed tools are trained on-line, using their ability for automatic deduction rules based on data, during model application for evaluation tasks. The model, consequently changes the data analysis stage, limits the scope of the necessary expertise in the area, where the assessment model can be used and, to some extent, the shape of the model becomes independent from the current range of available data. These features increase its ability to generalize and to cope with the data of previously undefined classes, as well as improve its resistance to gaps occurring in the data. Performance of the model presented in this paper is tested and verified on the basis of real-life data, which would resemble a potentially real practical application. Preliminary tests performed within the scope of this work indicate that the developed model can form a starting point for further research as some of the used mechanisms have a fairly high efficiency and flexibility.

Keywords: risk management, scoring model, information theory.

APPLIED SCORING MODELS

This article is intended to present an innovative and flexible approach to *scoring model construction.* The term scoring model used within this paper should be understood as a generic tool enabling to apply a risk mark to a given object, provided that the set of objects of the same time with observed risk results was previously collected. The assembled set of observations would be referred to as a historical data. The risk mark is an indicator, which permits to estimate at the moment the assessment of the future risk related to a given object. The key assumption of all scoring models is the existence of possibility to predict future risk on the basis of past characteristics of objects and observed results. This assumption is proved to be right by market practice, however, recently, as the market changes are becoming more intense and frequent, many models require experts adjustments, which would compensate their inertia.

Models allowing to predict future risk on the basis of historical data are fundamental tools applied for risk assessment and management within banks, insurance companies and other financial sector entities. They are applied to asses risks related to various contracts (e.g. loans, insurances, etc.) during the sales process and are intended to allow company to maximize its profit by managing the structure and margins of sold products. The most fundamental applications are:

- minimizing the number of customers or products with undesirable characteristics (i.e. to mit-

igate the number of borrowers which are most likely to default within the banking sector),
- proper evaluation of the risks associated with customer or product (i.e. offering the customer an adequate price for insurance products),
- maximization of profit realized by adjusting the margins to average levels of risk.

The model is construed in technical manner as a mathematical and informational tool, which allows to apply a numeric (continuous or discrete) measure to an object described by the collection of categorical and continuous variables. This measure should accurately reflect the future risk (understood here as a necessity to bear the cost) related to this objects. In most applications models are constructed to solve a particular entity of the problem. Currently, there are two main categories of risk scoring models:

- *basket model* – it is the most commonly used approach, based on market experience. It utilizes the parameters to divide objects into subgroups (portfolios). For each portfolio a likelihood of risk event is evaluated on the basis of the classic Laplace probability definition. To obtain the most accurate results the observations are divided also across the time line and different averaging techniques are adopted (e.g. moving averages, moving windows, trimmed means, etc.). This method owes its popularity to the simplicity of implementation and back testing. The most significant drawback of this method stems from the impossibility of changing classification parameters (which would ultimately result in changing the model). Moreover, this method does not permit to assess each object individually, therefore, we may observe changes in average quality of buckets along the timeline of model application;
- *function model* – the other popular approach is to attempt to build a function of object variables, which would map them directly to the probability of risk even occurrence. This method is less popular due to the need of developping a function model (e.g. to identify the character of each parameter impact or to eliminate cross-correlations), which is usually time-consuming and requires expertise in the area of application of the model. Moreover, the verification of the effectiveness of this method is much more difficult as it requires a proficiency in the field of statistics for determining the prediction power. The function model has significant advantages, of which the biggest is the ability to assign each object an individual score. Moreover, the use of the evaluation function allows to fully utilize the information carried by continuous variables without the necessity to divide values of such a variable into basket. Both popular models are similar to each other with respect to the steps required to implement and utilize them;
- *construction phase* – the analysis of collected historical data is performed, key variables are identified and applied to derive the model, the model is subsequently constructed and parametrized. This step is performed only once, at the beginning of the application of the model;
- *application phase* – this phase lasts as long as the model is applied. The new objects are assessed and the data on their performance are collected;
- *testing phase* – this phase periodically interrupts application phase. The actual observations are compared with attributes designated by the model, and results are utilized to improve parameters of the model.

The flow of models application, as indicated above means that they are not capable of fast adaptation to changing market conditions. Moreover there is no possibility to easily extend them to utilize additional data.

DEVELOPED SCORING MODEL

ASSUMPTIONS

In this article, a new, flexible approach to derive the scoring model construction is presented. The most important issue is to provide within the proposed model the capability to automatically detect dependencies between input data variables and observed risk events on the basis of collected samples. Such an approach shall result in versatility of the model and capability to work correctly for different instances of scoring problem. Moreover, it should allow to determine on-line, which parameters are the most valuable. The key assumptions, which shall be met by the developed model are:

- ability to assess the objects without being fitted to the particular instance of the problem,
- high utilization of the most current information,

- no requirement for periodic updates of model in order to achieve desirable quality of prediction,
- possibility of effective implementation,
- ability to use continuous and discrete variables,
- ability to work with data and a wide range of variables together with their values.

MODEL CONSTRUCTION

The model presented within this article is a hybrid solution based on artificial neural networks and basic concepts of machine learning and information theory (mainly decision trees). The most fundamental difference between classical models and the new approach is its *modus operandi*. Instead of creating strict classification rules on the basis of historical data and then applying them to evaluate new objects, the proposed model preselects the historical data sample and creates evaluation rules on-line for each object to be evaluated. Therefore the proposed scoring model is composed of two main parts:

- *the classifier module* – it is based on conditional entropy (see e.g. [1]) and allows to preselect a collections of historical data. Its main goal is to provide approximation with a collection of objects within the historical sample, which are the most similar to the one, being evaluated. Consequently its structure must allow to measure the distance between objects. This can be achieved by applying some form of a metric or by using the decision class building algorithm, such as e.g. decision trees. Our model uses an original semi-classification method (see chapter *Information gain based classifier*), due to its effectiveness to cope with large data sets. The designation of a set of similar cases in accordance with the metric might occur to be more accurate, but it would be also more time consuming which would deteriorate model effectiveness significantly. Direct application of the algorithms building decision trees, would also be not effective in the context of the model performance. Therefore, in this paper an original concept is presented, which eliminates the most critical problems. Namely, it utilizes the concept of information gain, which is in fact a difference between the entropy of a problem and conditional entropy calculated for selected parameters fixed. On the basis of this value a hierarchy of parameters is constructed. The classifier provides collection of the most similar cases by selecting objects to meet the sample group size by thereby assuming consistence of variables, starting from the most informative ones.
- *the approximator module* – it is based on multilayer perceptron (see e.g. [2]), which is further referred to as MLP. It is trained using the sample preselected by the classifier module and allows to score the new object. Its use in practice does not require expert knowledge on the issues, to assess which model is used. MLP in the premise should allow to assign a new object attributes by the use of standard procedures for learning and use of the network. For any new patterns, a set of historical data is provided by a classifier. An important issue is that despite the use of the classifier, the network is trained with the full set of parameters. Moreover, unlike classifier, the MLP adopts for continuous variables their actual values instead of the class to which they have been assigned by the classifier.

The classifier module is a preselection tool, which generates learning sample for approximator module. It must possess knowledge of data structure, however it can be refreshed periodically or on demand (depending on actual business needs) without losing its original capabilities. The approximator module is trained for each evaluated object separately, which in turn permits to alter its structure in any given moment to utilize all of the available data and knowledge.

MODEL OPERATING SCHEMA

The generated model is a tool implementing the following operating schema:

1. The preliminary phase of preparation of the model to deal with a particular type of data and calculating the information gain. It may be split into two key activities:
   - data analysis sub-phase – the model analysis for data structure – it requires user to decide which numeric variables should be treated as continuous and which as categorical. The result of this analysis is a dictionary of parameters and attributes, their respective ranges of values and the number of categories. Rebuilding the dictionary should not be performed in the operating model, however, it can easily be extended during the use of the model to reflect the changes in possible values or range of available variables.

- information gain calculation sub-phase – for all parameters there are applied methods of performing division parameters into classes and enumerators. In sequel appropriate information gain parameters are evaluated. The effect of this phase is a structure of information about the parameters, their possible values, and their informational power. This phase may be repeated with each addition of new observations into the historical data set.

2. Phase of determining the structure of tools. In this step, based on the data collected in phase 1, information about the characteristics of the problem determines the structure of used tools:
   - classifier – must work according to the data structure collected and analyzed in subphase (a) of phase (1).
   - approximator – must be able to work on the data provided by classifier.
3. Phase application of the model. Within this step a model is used to evaluate new objects. The new observations are included into the historical data and model is able to utilize them to make new predictions. It is also required to periodically verify the accuracy of the model. The model may be extended to be capable of including information on the new parameters and the structure of tools may also be adjusted to provide higher accuracy.

THEORETICAL FUNDAMENTALS OF MODEL

INFORMATION GAIN BASED CLASSIFIER

Entropy

Entropy in the context of this work is determined for discrete variable and should be understood as proposed by Shannon [3]. Under the present proposal, entropy is defined as a measure of uncertainty and the weighted average amount of information carried by a single value of a random variable. The weight is assumed for the probability of adoption by the value of a random variable. In the sense of Shannon entropy, denoted by the symbol H for a discrete random variable X with set of admissible values $\chi = \{x_1, ..., x_n\}$ of the probability function $P(x)$ is described by the formula:

$$H(X) = -\sum^{n} P(x_i) \log_b P(x_i).$$

Conditional entropy

Conditional entropy is an extension of the concept of entropy for systems with a larger number of random variables. In this paper, the term is used in accordance with the definition given in [1]. Conditional entropy is a measure of the entropy of an unknown discrete random variable Y at a fixed value of a discrete random variable X. An important property of conditional entropy is irrelevance to determine whether the variables X and Y are dependent or not. If by the $H(Y|X = x)$ we assume the entropy of Y conditioned by the variable X taking a fixed value of x, then $H(Y|X)$ is the result of averaging the values of $H(Y|X = x)$ on the set of all possible values that x takes on a set X. Assume, therefore that X is a discrete random variable with support X, then the conditional entropy $H(Y|X)$ is described by the formula:

$$IG(Y, X) = H(Y) - H(Y|X).$$

Information gain

The information gain in this work is understood as a concept in the field of machine learning, as defined in [4]. In general, the term describes the change in entropy of information between the initial state and the state in which shall be selected for a given parameter:

Application of information gain in machine learning

A measure of the information gain in classical machine learning (see e.g. [4]) is used to iteratively select the best variable to split in the construction of decision trees [1]. An example of a popular algorithm implements the Information Gain Algorithm ID3 [5]. Classifier proposed in this paper, as well as algorithms for construction of decision trees, are based on the value of the information gain. The main difference between the decision trees and the proposed classifier is that it takes into account only the information designated parameter increment between the input state, and a state in which a single parameter has a fixed value.

In the proposed model, the classifier calculated values of the parameter increment of information allow to build a hierarchy of significance P parameters. We define a set of parameters PH

objects as a collection of the following pairs (p_i, h_i), where $p_i \in P$ is the identifier of the parameter, and h_i is calculated for the growth parameter information. The classifier implements the steps:

1. Make a set of decision rules for an object PW_o as the empty set. Its elements are pairs of the form (p_i, v_i), where $p_i \in P$ is the ID parameter and w_i is the value.
2. Choose the most important parameter p_j such that $h_j > h_i \forall (p_i, h_i) \in PH$.
3. Check the w_o of the newly tested object parameter p_i.
4. Determine the cardinality l of the set of historical elements for which the value of the parameter p_j is equal w_o and the corresponding values w_i of all the parameters p_i belonging to PW_o.
5. Remove pair (p_j, h_j) from the set of PH.
6. Check if the cardinality l satisfies the assumptions determined by business requirements for given instance of a problem. If so, add a pair (p, w_o) to set PW_o and exit. If not, depending on the reasons for non-compliance with the rules add or no pair (p_j, w_o) to set PW_o and repeat steps 2–6.

Resolved decision trees issues

Practical application of decision tree learning methods has allowed the identification of a number of the problems described in [4]. The proposed approach allowed to mitigate their impact on model:

- over development of decision trees, which often reduces the ability of generalization, is mitigated by adapting strict boundaries for result,
- missing exclusion of parameter identifying individual objects. Such problem can occur in the case of numeric variables that are not defined as direct object identifiers (such as social security numbers, bank accounts, credit cards). This attribute will automatically minimize the entropy, and therefore automatically becomes the first parameter dividing set. However, the application automatically excludes the possibility of using it, as the result would not allow to generate sample of desired size,
- lack of using of continuous attributes, which stems directly from the lack of a direct generalization of the definition of entropy for a continuous variables. This problem is partial resolved by applying procedure as described in next chapter.

Continuous variables in classifier

The problem of handling continuous variables is partially resolved within this model by application of tool splitting data into baskets. The method used involves the designation of the optimal width of baskets by designating their boundaries according to the method proposed in [6]. This method assumes that the observed values are derived from the frequency of unknown function that generates an observable event. These events are forming baskets, which is cardinality determined to generate a histogram for the observed values. The problem is therefore to find the optimal number of baskets, which in practice is equivalent to finding the optimal width of the basket. The question of optimality can be described as minimizing MISE (integral mean-square error) between estimator in the form of a histogram and a priori unknown function of the frequency. Optimization is not possible to be directly applied because of the lack of knowledge of the real function of frequency. However, the method proposed in [6] allows the estimation and minimization of MISE based on the data.

The following procedure allows to find the optimal width of the bin for the histogram denoted as Δ_{opt}:

1. Data from the sample T are divided into N bins, each of which has a length of $\Delta = T/N$. The number of events collected in *i-th* interval, denoted as k_i is determined and the average and variance of number of events are calculated using the formulas:

$$k = \frac{1}{N}\sum_{i=1}^{N} k_i,$$

$$v = \frac{1}{N}\sum_{i=1}^{N} (k_i - k)^2.$$

2. The value of the cost function is calculated as:

$$C(\Delta) = \frac{2k - v}{\Delta^2}.$$

3. Repeat steps 1–2 changing N to obtain a value Δ_{opt}, which minimizes the value of $C(\Delta)$.

The obvious problem associated with optimizing the width of the bin in this method is a selection of a test number of N baskets. The most correct approach would be to examine all the possible numbers of baskets. Such an approach, especially for large sample size would be very aggravating in terms of performance. Therefore, the

model used is an approach that narrows the range of possible frequencies baskets. This scheme is based partially on the rule for the optimum number of baskets histograms proposed in [7]:

$$N_{opt} = [\log_2 n + 1].$$

Struges formula (see e.g. [7]), formula allows to directly determine the optimal number of baskets, but it works correctly for the parameters of a normal distribution. Therefore, the primary tool used in the model evaluation procedure is proposed in [6], and model is used to narrow down the searches for methods to minimize the estimated error. The model is therefore tested for the values of the $N \in [0.5 \times N_{opt}, 1.5 \times N_{opt}]$ and $N \in \boldsymbol{N}$.

MLP APPROXIMATOR

Artificial neuron

McCulloch-Pitts neuron model (see in [8]) is a very precise and clear description of the model of artificial neuron, but its major limitation is narrowing the number of possible outcomes for the two binary values (0,1). For approximator problem there is a need to use a more general model of the neuron. According to [2], for the construction of multilayer perceptron linear neurons can be used. A generalized model of the neuron consists of two components:

- adder, which sums the weighted input values and an additional bias. The principle operation of the adder in a generalized neuron can be defined using the following formula:

$$x = \sum_{i=1}^{m} x_i w_i + \theta,$$

where: $[v_1, w_2, ..., v_m]$ – the weight vector of inputs,
$[x_1, x_2, ..., x_n]$ – a vector of input neuron,
Θ – the value of bias for the neuron,

- activation function, which takes as input a designated sum. Activation function neuron output amplitude narrows the range [0, 1], or alternatively [–1, 1]. Such a function should satisfy two conditions:
  - continuity between its minimum and maximum values, otherwise it would be inappropriate to assume that the output of the neuron would be obtained for all inputs,
  - continuous derivative, preferably possible to determine in an analytical form This assumption is needed to make the application of learning algorithms based on the concept of gradient minimization of cost function was possible.

The choice of the activation function changes the operating characteristics of artificial neuron model. Due to the properties (particularly easy to determine the derivative) one of the most commonly used activation function is the sigmoidal function (see e.g. [9]). The model allows to use two types of sigmoidal functions:

- *Unipolar sigmoid function.* This function is described by the formula: $\phi(x) = \frac{1}{1+e^{-\ betax}}$. This function gives for strongly positive input values close to 1 and the output value approaches 0 when input assumes a negative value. In the vicinity of 0 the function takes values close to 0.5. The β regulates the steepness of the function,
- *Bipolar sigmoid function.* This function is described by the formula: $\phi(x) = \frac{2}{1+e^{-\beta x}} - 1 = \frac{1-e^{-\beta x}}{1+e^{-\beta x}}$. It is very similar to unipolar activation function, but the output value close to –1 is taken when input assumes a negative value. In the vicinity of 0 the function takes values close to 0. As in the unipolar activation function, parameter β regulates the steepness of the function.

Multilayer Perceptron

The basic tool used in the approximation is multilayer perceptron (see e.g. [10]), which is a variant of artificial neural network. It allows one to classify, or assign to a certain set of input parameters a baseline value of a specific range. Multilayer perceptron model is built of k layers of neurons. Between neurons in the same layer there are no connections. Connections in a multilayer perceptron occur only between two adjacent layers, wherein the connections are complete and unidirectional. This means that each neuron of layer $i + 1$ as input takes the values of all outputs of neurons of layer *i.* In the multilayer perceptron model, there are no connections to the higher and lower layers.

MLP ability to approximate non-linear functions

According to the theorem of universal approximation property (see e.g. [11]), an MLP network of feed-forward type with a single hidden layer containing a specified number of neurons with arbitrary activation functions is a universal

approximator within the space $C(\boldsymbol{R}^M)$. This result is extremely important because it guarantees, in the context of this work, that the multilayer perceptron can approximate any function mapping input parameters to the probability of the event, if such a function exists.

Kurt Hornik [12] showed that a proper selection of the size of the MLP neural network with a single hidden layer allows the network to become a universal approximator. It is worth noting that one of the first versions of Cybenko Theorem proved in 1989 for sigmoid activation function refers to the capacity of approximation by MLP. This theorem, moreover imposes no restrictions on the number of neurons in the output layer, which is why we present the theorem in the form of a single neuron in the output layer.

Theorem *(Universal approximation theorem)*

Let ϕ be not fixed, limited and monotonically increasing continuous function of activation. Let $U \subseteq \mathbb{R}^n$, and let U be a compact set. Then $\forall f \in C(U)$, (where by $C(U)$ we mean the space of continuous functions f: $U \subseteq \mathbb{R}^n \to \mathbb{R}$) $\forall \epsilon > 0\, \exists n \in N$,) and there are adequate w_{ij}, θ_i, w_i where $i \in \{1, ..., n\}$ and $j \in \{1, ..., m\}$, such that:

$$(A_n f)(x_1, ..., x_m) = \sum_{i=1}^{n} w_i \phi \left(\sum_{j=1}^{m} x_i w_{ij} + \theta_i \right),$$

is a uniform approximator of function f:

$$\|f - A_n f\| < \epsilon,$$

holding for all $(x_1, ..., x_n)$ from the domain.

In the presented claim approximating function is described as $A_n f$. This notation is to emphasis the added that the size of the hidden layer plays an important role.

MLP learning with back propagation procedure

The back propagation algorithm (see e.g. [13]) main goal is to build a map that allows to return value for each pattern as close as possible the expected value of activation. This mapping, in accordance with [13] is constructed on the basis of a reference sequence $P^{(j)} = (X^{(j)}, Z^{(j)})$, where $1 \leq j \leq N$ is the number pattern. Within the reference pair $X^{(j)}$ represents the input vector, and $Z^{(j)}$ a vector of expected outputs from the network. After completion of the process of learning the expected result at the output of the network, denoted $Y^{(j)}$ for $X^{(j)}$ is $Z^{(j)}$ which is the result obtained by the network should be equal to the result expected $Y^{(j)} = Z^{(j)}$ Therefore, if an error is introduced based on minimizing the error-square average:

$$Q = \sum_{j=1}^{N} Q^{(j)} = \frac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{M} \left(z_i^{(j)} - y_i^{(j)} \right)^2,$$

where: $z_i^{(j)}$ is the expected output of the i-th output neuron in the pattern j, and $y_i^{(j)}$ is the corresponding value determined at the output of the network.

Finding the best possible network weights is therefore equivalent to the determination of the global minimum of Q. Determining the direction of the search is done by determining the gradient $\nabla Q = \frac{\delta Q}{\delta}$ and, therefore, following the steepest descent direction. In the case of a single neuron, this task could be performed directly, by having full information about the expected value of the output. In the case of a multilayer network is not possible to know directly the expected output values for each neuron. Therefore, in accordance with [13] applying the concept of determining the chain of gradients is necessary.

Back-propagation method is based on gradient search, which results in convergence to the nearest minimum. Therefore it may not guarantee to achieve a global minimum. This method is sensitive to the choice of the initial conditions. Furthermore, back propagation method may not converge to a local minimum, oscillating around it instead.

Another problem highlighted in [13], is the issue of partial derivatives counter intuitive impact on the length of the step. If the error function is relatively flat , then the derivative assumes a small value, which results in the short step, and therefore on the flat areas of the error correction function is done by weight relatively slowly. In the situation when the error function is steep, derivatives assume higher values, which automatically results in an increase in step length. In order to reduce such behavior method introduces the concept of momentum, which allows to include information on previous modifications into current step. This technique allows in many cases to improve the effectiveness of the back propagation method, although there is a method which guarantees or accelerating convergence in any case.

An important problem is also an initialization of weights. The key issue is to ensure they are

non-zero and their maximum possible asymmetry. In practice, a solution is based on a random selection of weights, which significantly reduces the risk of obtaining too similar weights in the network.

Multilayer Perceptron is trained by providing a complete sequence of patterns, which are presented iteratively and also iteratively adjusted weights are the network. Within the approximator model following stopping criteria for learning are applied:

- the maximum number of iterations condition,
- increase the value of the error condition – the algorithm stops if the error after the presentation of each sequence increases (which serves as a protection against oscillation).

TESTS

On the basis of theoretical results presented above a software testing platform is developed within the scope of work on this paper. The data set used to test the model was taken from the site www.kaggle.com, where a "claim prediction challenge" for the insurance company has been published. The tests procedure was as follows:

1. Selecting one item from a set of verification, selecting the matching elements from the training set (classification phase) and, in addition, drawing from a set of verification pot of events according to the same criteria.
2. Performing the complete approximation phase.
3. Calculation of the basic statistics for observations in both the training and averification set.
4. Comparing stats with the results of the approximator.

During the tests the model behaviour is dependant to sample sizes and distribution of cases, for which the necessity to pay compensation occured. Below we present sample results obtained during testing phase:

1. Sample of size of 423 elements, including 3 cases, for which the necessity to pay compensation occured (0.866%). Within the verification set there were 231 elements, including 2 cases, for which the necessity to pay compensation occured (0.709%):
   - for the 2 elements with compensations the average network result amounted 0.00748706 with standard deviation of 0.00234744;
   - for the 229 elements without compensations the average network result amounted 0.00760141 with standard deviation of 0.00382598,

   This example confirms that model may fail to properly distinct elements of higher and lower risk.
2. Sample of size of 897 elements, including 7 cases, for which the necessity to pay compensation occured (0.780%). Within the verification set there were 1061 elements, including 7 cases, for which the necessity to pay compensation occured (0.660%):
   - for the 7 elements with compensations the average network result amounted 0.00586282 with standard deviation of 0.00000000,
   - for the 1054 elements without compensations the average network result amounted 0.00565989 with standard deviation of 0.00092168,

   In the latter example presents the situation in which the model was able to distinct elements of higher and lower risk.

CONCLUSIONS

On the basis of tests, we concluded that the developed evaluation model cannot be applied in its current form in practice. However, the results are promising and suggest that the adopted approach could efficiently be used in practice subject to more profound research, especially within the approximator module. A key problem is the lack of comparability of results of approximator between instances of the problem and the tendency to return a fixed outcome for each input (this could be mitigated by applying penalty function for zero weights). This problem occurs especially for collections in which the zero weights are strong local minimum. An important observation is that the classifier used in the current model is a tool of relatively high accuracy and is properly fulfilling its role.

The data used to test this model are very difficult to analyze, mainly due to a very low share of cases for which adverse events occurred and their fairly distribution over the data set. The identification of rules for the occurrence of adverse events is a complicated issue. The scoring model proposed in this paper is capable,

provided the tools are properly parametrized, to identify within homogeneous groups of observations (within the meaning of the classifier) cases with potentially greater risks, which entitles the statement that this model could potentially be developed into a functional tool for risk assessment. The scoring model presented in this paper may therefore be the starting point for further research into this type of approach to the construction of assessment models.

REFERENCES

1. Cover T.M., Thomas J.A.. Elements of Information Theory. Wiley, 1991.
2. Pal S.K. and Mitra S. Multi-layer perceptron, fuzzy sets and classification. IEEE Transactions on Neural Networks, 3, 1992, 683-697.
3. Shannon C. A mathematical theory of communication. Bell System Technical Journal, 27(3), 1948, 379-423.
4. Mitchell T. Machine Learning. McGraw-Hill, 1997.
5. Quinlan J.R. Induction of decision trees. Machine Learning, 5(1), 1986, 71-100.
6. H. Shimazaki, Shinomoto S. A method for selecting the bin size of a time histogram. Neural Computation, 19, 2007, 1503-1527.
7. Sturges H.A. The choice of a class interval. Journal of the American Statistical Association, 1926, 65-66.
8. McCulloch W.S., Pitts W.H. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 1943, 115-133.
9. Hassoun M.H. Fundamentals of Artificial Neural Networks. The MIT Press, 1995.
10. Rosenblatt F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan, 1962.
11. Csaji B.C. Approximation with artificial neural networks. http: //citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.2647&rep=repl&type=pdf, 2001.
12. Hornik K. Approximation capabilities of multilayer feedforward networks. Neural Networks, 4, 1991, 251-257.
13. Riedmiller M. Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms, http://citeseerx.ist.psu. edu/viewdoc/download?doi=10.1.1.27.7876&rep=repl&type=pdf.